## PCS 2428 / PCS 2059
## Inteligência Artificial

Prof. Dr. Jaime Simão Sichman
Prof. Dra. Anna Helena Reali Costa

Jogos

## Jogos: considerações gerais

- Aplicações atrativas para métodos IA desde o início.
  - Formulação simples do problema (ações bem definidas)
  - Ambiente totalmente observável (geralmente);
  - Abstração (representação simplificada de problemas reais);
  - Sinônimo de "inteligência";
    - Primeiro algoritmo para xadrez foi proposto por Claude Shannon na década de 50.
- Porém desafiador:
  - Tamanho + limitação de tempo ($35^{100}$ nós para xadrez);
  - Incerteza devido ao outro jogador;
  - Problema "contingencial": agente deve agir antes de completar a busca

2

## Formulando e resolvendo o problema

- 2 jogadores, revezam o lance, são adversários
- Formulação
  - **Estado inicial**: posições do tabuleiro + de quem é a vez
  - **Estado final**: posições em que o jogo acaba
  - **Operadores**: jogadas legais (=válidas)
  - **Função de utilidade**: valor numérico do resultado (pontuação)
- Busca: algoritmo minimax
  - **Idéia**: **maximizar a utilidade** (ganho) supondo que o adversário vai tentar minimizá-la.
  - Minimax faz **busca cega em profundidade**.
  - O agente é MAX e o adversário é MIN.

3

## Algoritmo Minimax

© Russel and Norvig, AIMA slides

```
function MINIMAX-DECISION(state) returns an action
   inputs: state, current state in game
   return the a in ACTIONS(state) maximizing MIN-VALUE(RESULT(a, state))

function MAX-VALUE(state) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← −∞
   for a, s in SUCCESSORS(state) do v ← MAX(v, MIN-VALUE(s))
   return v

function MIN-VALUE(state) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← ∞
   for a, s in SUCCESSORS(state) do v ← MIN(v, MAX-VALUE(s))
   return v
```

5

## Minimax

- Passos:
  - Gera a árvore **inteira** até os estados terminais (ganha, perde ou empata).
  - Aplica a **função de utilidade** nas folhas.
  - Propaga os valores dessa função subindo a árvore através do minimax.
  - Determinar qual a ação que será escolhida por MAX.

Jogada decidida por MAX
MAX
MIN
3
A1 A2 A3
3 2 2
A11 A12 A13 A21 A22 A23 A31 A32 A33
3 12 8 2 4 6 14 5 2

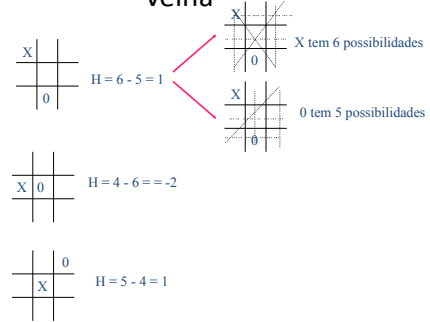
6

## Críticas

- Problemas
  - Tempo gasto para determinar a decisão ótima é totalmente impraticável (ir até as folhas), porém o algoritmo serve como base para outros métodos mais realísticos.
  - Complexidade: O($b^m$) – idem Busca em Profundidade.
- Para melhorar
  1) Limitar a profundidade da busca e substituir função de utilidade por função de avaliação (heurística);

  2) Podar a árvore onde a busca seria irrelevante: poda alfa-beta

7

## Função heurística para o jogo da velha

8

## Uso da Função de Avaliação

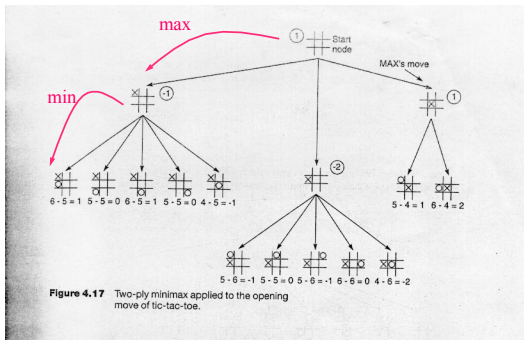



**Figure 4.17** Two-ply minimax applied to the opening move of tic-tac-toe.

## Poda Alpha-Beta

- Objetivo: não expandir desnecessariamente nós durante o minimax.
- Idéia: não vale a pena piorar, se já achou algo melhor.
- Mantém 2 parâmetros:
  - $\alpha$ – melhor valor (no caminho) para MAX
  - $\beta$ – melhor valor (no caminho) para MIN
- Teste de expansão:
  - $\alpha$ não pode diminuir (não pode ser menor que um ancestral)
  - $\beta$ não pode aumentar (não pode ser maior que um ancestral)

10

## Poda Alpha-Beta

© Russel and Norvig, AIMA slides

```
function ALPHA-BETA-SEARCH(state) returns an action
   inputs: state, current state in game

   v ← MAX-VALUE(state, −∞, +∞)
   return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value
   inputs: state, current state in game
           α, the value of the best alternative for MAX along the path to state
           β, the value of the best alternative for MIN along the path to state

   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← −∞
   for a, s in SUCCESSORS(state) do
      v ← MAX(v, MIN-VALUE(s, α, β))
      if v ≥ β then return v
      α ← MAX(α, v)
   return v
```

## Poda Alpha-Beta

© Russel and Norvig, AIMA slides

```
function MIN-VALUE(state, α, β) returns a utility value
   inputs: state, current state in game
           α, the value of the best alternative for MAX along the path to state
           β, the value of the best alternative for MIN along the path to state

   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← +∞
   for a, s in SUCCESSORS(state) do
      v ← MIN(v, MAX-VALUE(s, α, β))
      if v ≤ α then return v
      β ← MIN(β, v)
   return v
```

## Poda Alpha-Beta: exemplo – II

Continua expansão de B : folha com 12 > 3 (β não muda seu máximo), depois folha com 8>3 (β não muda seu máximo). B não tem mais filhos, portanto $\beta_1 = 3$ e α do pai pode ser iniciado. Continua expansão, com α limitando a busca.

α ≥ 3

MAX

$\beta_1 = 3$ B $\beta_2 \leq +\infty$ $\beta_3 \leq +\infty$

MIN

3 12 8 2 4 6 14 5 2

*Valor de, no mínimo... (só pode aumentar)*

15

Exercício
Decidir a jogada de MAX (A, B ou C) considerando as utilidades fornecidas nas folhas. Adotando a poda alfa-beta, indicar quais arestas/subárvores serão podadas.
MAX
MIN
MAX
MIN
A
B
C
D
E
F
G
H
I
J
K L M N O P Q R S T U
2 3 5 7 0 5 7 2 1 5 7
19

Exercício
MAX
MIN
MAX
MIN
[ -∞, +∞]
20

Exercício
[ -∞, +∞]
[ -∞, +∞]
21

Exercício
[ -∞, +∞]
[ -∞, +∞]
[ -∞, +∞]
22

Exercício
[ -∞, +∞]
[ -∞, +∞]
[ -∞, +∞]
23

Exercício
[ -∞, +∞]
[ -∞, +∞]
[ -∞, +∞]
2
24

Exercício
MAX
MIN
MAX
MIN
A
[ -∞, +∞]
B
[ -∞, 3]
C
D
[ 3, +∞]
E
[ -∞, 3]
F
G
H
I
J
K L M N O P Q R S T U
2 3 5 7 0 5 7 2 1 5 7

Exercício
MAX
MIN
MAX
MIN
A
[ -∞, +∞]
B
[ -∞, 3]
C
D
[ 3, +∞]
E
[ -∞, 3]
F
G
H
I
J
K L M N O P Q R S T U
2 3 5 7 0 5 7 2 1 5 7

Exercício
MAX
MIN
MAX
MIN
A
[ -∞, +∞]
B
[ -∞, 3]
C
D
[ 3, +∞]
E
[ -∞, 3]
F
G
H
I
J
5
K L M N O P Q R S T U
2 3 5 7 0 5 7 2 1 5 7

Exercício
MAX
MIN
MAX
MIN
A
[ -∞, +∞]
B
[ -∞, 3]
C
D
[ 3, +∞]
E
[ -∞, 3]
F
G
H
I
J
5 ≥ 3 – Poda β
K L M N O P Q R S T U
2 3 5 7 0 5 7 2 1 5 7

Exercício
MAX
MIN
MAX
MIN
A
[ -∞, +∞]
B
[ -∞, 3]
C
D
5
[ 3, +∞]
E
[ -∞, 3]
F
G
H
I
J
K L M N O P Q R S T U
2 3 5 7 0 5 7 2 1 5 7

Exercício
MAX
MIN
MAX
MIN
A
[ -∞, +∞]
3
B
[ -∞, 3]
C
D
[ 3, +∞]
E
[ -∞, 3]
F
G
H
I
J
K L M N O P Q R S T U
2 3 5 7 0 5 7 2 1 5 7

Exercício
MAX
[ 3, +∞]
A
MIN
B
[ -∞, 3]
C
[ 3, +∞]
D
[ 3, +∞]
MAX
E
[ 3, +∞]
F
[ -∞, 3]
G
[ 3, +∞]
H
I
[ 3, +∞]
J
MIN
K L M N O P Q R S T U
2 3 5 7 0 5 7 2 1 5 7
49

Exercício
MAX
[ 3, +∞]
A
MIN
B
[ -∞, 3]
C
[ 3, +∞]
D
[ 3, +∞]
MAX
E
[ 3, +∞]
F
[ -∞, 3]
G
[ 3, +∞]
H
I
[ 3, +∞]
J
1
MIN
K L M N O P Q R S T U
2 3 5 7 0 5 7 2 1 5 7
50

Exercício
MAX
[ 3, +∞]
A
MIN
B
[ -∞, 3]
C
[ 3, +∞]
D
[ 3, +∞]
2
MAX
E
[ 3, +∞]
F
[ -∞, 3]
G
[ 3, +∞]
H
I
[ 3, +∞]
J
MIN
K L M N O P Q R S T U
2 3 5 7 0 5 7 2 1 5 7
51

Exercício
MAX
[ 3, +∞]
A
MIN
B
[ -∞, 3]
C
[ 3, +∞]
D
[ 3, +∞]
2 ≤ 3 – Poda α
MAX
E
[ 3, +∞]
F
[ -∞, 3]
G
[ 3, +∞]
H
I
[ 3, +∞]
J
MIN
K L M N O P Q R S T U
2 3 5 7 0 5 7 2 1 5 7
52

Exercício
MAX
[ 3, +∞]
A
2
MIN
B
[ -∞, 3]
C
[ 3, +∞]
D
[ 3, +∞]
MAX
E
[ 3, +∞]
F
[ -∞, 3]
G
[ 3, +∞]
H
I
[ 3, +∞]
J
MIN
K L M N O P Q R S T U
2 3 5 7 0 5 7 2 1 5 7
53

Exercício
Jogada escolhida
MAX
[ 3, +∞]
MIN
B
[ -∞, 3]
C
[ 3, +∞]
D
[ 3, +∞]
MAX
E
[ 3, +∞]
F
[ -∞, 3]
G
[ 3, +∞]
H
I
[ 3, +∞]
J
MIN
K L M N O P Q R S T U
2 3 5 7 0 5 7 2 1 5 7
54